Intersection of Cylinders

David Eberly, Geometric Tools, Redmond WA 98052
https://www.geometrictools.com/

Created: November 21, 2000
Last Modified: April 16, 2017

Contents

1 Introduction

This document shows how to determine whether two bounded cylinders intersect. The algorithm uses the method of separating axes, although the construction is more complicated than one encounters when separating convex polyhedra. The resulting algorithm is a fairly expensive one. If you plan on using cylinders for bounding volumes in a real-time graphics engine—think twice. A better alternative to a cylinder is a *capsule*, the set of points a specified distance from a line segment. Two capsules intersect if and only if the distance between capsule line segments is smaller or equal to the sum of the capsule radii, a much cheaper test to perform.

2 Representation of a Cylinder

A cylinder has a center point $\mathbf{C}$, unit-length axis direction $\mathbf{W}$, radius r and height h. The end disks of the cylinder are centered at $\mathbf{C} \pm (h/2)\mathbf{W}$. Let $\mathbf{U}$ and $\mathbf{V}$ be any unit-length vectors for which $\{\mathbf{U}, \mathbf{V}, \mathbf{W}\}$ is a right-handed set of orthonormal vectors. That is, the vectors are unit length, mutually orthogonal, and $\mathbf{W} = \mathbf{U} \times \mathbf{V}$. Points in the cylinder are parameterized by

$$\mathbf{X}(\theta, t) = \mathbf{C} + (s\cos\theta)\mathbf{U} + (s\sin\theta)\mathbf{V} + t\mathbf{W}, \;\; \theta \in [0, 2\pi), \; 0 \le s \le r, \; |t| \le h/2 \tag{1}$$

The projections of a cylinder onto a line are determined solely by the cylinder wall, not the end disks.

The choice of $\mathbf{U}$ and $\mathbf{V}$ is arbitrary. Intersection queries between cylinders should be independent of this choice, but some of the algorithms are better handled if a choice is made. A quadratic equation that represents the cylinder wall is $(\mathbf{X} - \mathbf{C})^{\mathsf{T}}(I - \mathbf{W}\mathbf{W}^{\mathsf{T}})(\mathbf{X} - \mathbf{C}) = r^2$. The boundedness of the cylinder is specified by $|\mathbf{W} \cdot (\mathbf{X} - \mathbf{C})| \le h/2$. This representation is dependent only on $\mathbf{C}$, $\mathbf{W}$, r, and h.

3 Nonintersection of Convex Objects by Projection Methods

Consider the problem of determining whether two convex objects in 3D are intersecting. This *test-intersection* geometric query is concerned only about whether the objects intersect, not about where they intersect. The latter problem is said to be a *find-intersection* geometric query. This document is about the test-intersection query for two bounded cylinders.

3.1 Separation by Projection onto a Line

A test for nonintersection of two convex objects is simply stated: If there exists a line for which the intervals of projection of the two objects onto that line do not intersect, then the objects do not intersect. Such a line is called a *separating line* or, more commonly, a *separating axis*. The translation of a separating line is also a separating line, so it is sufficient to consider lines that contain the origin. Given a line containing the origin and with unit-length direction $\mathbf{D}$, the projection of a compact convex set C onto the line is the interval

$$I = [\lambda_{\min}(\mathbf{D}), \lambda_{\max}(\mathbf{D})] = [\min\{\mathbf{D} \cdot \mathbf{X} : \mathbf{X} \in C\}, \max\{\mathbf{D} \cdot \mathbf{X} : \mathbf{X} \in C\}]$$

Two compact convex sets C_0 and C_1 are separated if there exists a direction $\mathbf{D}$ such that the projection intervals I_0 and I_1 do not intersect, $I_0 \cap I_1 = \emptyset$. Specifically they do not intersect when

$$\lambda_{\min}^{(0)}(\mathbf{D}) > \lambda_{\max}^{(1)}(\mathbf{D}) \text{ or } \lambda_{\max}^{(0)}(\mathbf{D}) < \lambda_{\min}^{(1)}(\mathbf{D}). \tag{2}$$

The superscripts correspond to the indices of the convex set. Although the comparisons are made where $\mathbf{D}$ has unit length, the comparison results are invariant to changes in length of the vector. This follows from $\lambda_{\min}(t\mathbf{D}) = t\lambda_{\min}(\mathbf{D})$ and $\lambda_{\max}(t\mathbf{D}) = t\lambda_{\max}(\mathbf{D})$ for $t > 0$. The Boolean value of the pair of comparisons is also invariant when $\mathbf{D}$ is replaced by the opposite direction $-\mathbf{D}$. This follows from $\lambda_{\min}(-\mathbf{D}) = -\lambda_{\max}(\mathbf{D})$ and $\lambda_{\max}(-\mathbf{D}) = -\lambda_{\min}(\mathbf{D})$. When $\mathbf{D}$ is not unit length, the intervals obtained for the separating axis tests are not the projections of the object onto the line; rather, they are scaled versions of the projection intervals. I make no distinction in this document between the scaled projection and regular projection. I will also use the terminology that the direction vector for a separating axis is called a *separating direction* and is not necessarily unit length.

For a pair of convex polyhedra, only a finite set of direction vectors needs to be considered for separation tests. That set includes the normal vectors to the faces of the polyhedra and vectors generated by a cross product of two edges, one from each polyhedron. I am aware of no general theory for constructing the smallest set of potential separating directions for other convex objects.

In Equation (2), allowing equality means that the two convex objects might be separated, but they also might be just touching. The intersection set does not have volume to it, so one might say that the objects are not overlapping. In this document, I will allow the equality because it allows for some simplification of the algorithm for separation testing. If you need strict separation, the algorithm may be modified appropriately to support this.

3.2 Projection of a Cylinder onto a Line

Let the line be $\lambda\mathbf{D}$ where $\mathbf{D}$ is a nonzero vector. The projection of a cylinder wall point onto the line is

$$\lambda(\theta, t) = \mathbf{D} \cdot \mathbf{X}(\theta, t) = \mathbf{D} \cdot \mathbf{C} + (r\cos\theta)\mathbf{D} \cdot \mathbf{U} + (r\sin\theta)\mathbf{D} \cdot \mathbf{V} + t\mathbf{D} \cdot \mathbf{W} \tag{3}$$

The interval of projection has endpoints determined by the extreme values of this expression. The maximum value occurs when all three terms involving the parameters are made as large as possible. The t-term has a maximum of $(h/2)|\mathbf{D} \cdot \mathbf{W}|$. The θ-terms, not including the radius, can be viewed as a dot product $(\cos\theta, \sin\theta) \cdot (\mathbf{D} \cdot \mathbf{U}, \mathbf{D} \cdot \mathbf{V})$. This is maximized when $(\cos\theta, \sin\theta)$ is in the same direction as $(\mathbf{D} \cdot \mathbf{U}, \mathbf{D} \cdot \mathbf{V})$. Therefore,

$$(\cos\theta, \sin\theta) = \frac{(\mathbf{D} \cdot \mathbf{U}, \mathbf{D} \cdot \mathbf{V})}{\sqrt{(\mathbf{D} \cdot \mathbf{U})^2 + (\mathbf{D} \cdot \mathbf{V})^2}}$$

and the maximum projection value is

$$\lambda_{\max} = \mathbf{D} \cdot \mathbf{C} + r\sqrt{|\mathbf{D}|^2 - (\mathbf{D} \cdot \mathbf{W})^2} + (h/2)|\mathbf{D} \cdot \mathbf{W}| \tag{4}$$

The construction uses the identities

$$\begin{aligned} &\mathbf{D} = (\mathbf{D} \cdot \mathbf{U})\mathbf{U} + (\mathbf{D} \cdot \mathbf{V})\mathbf{V} + (\mathbf{D} \cdot \mathbf{W})\mathbf{W} \\ &(\mathbf{D} \cdot \mathbf{U})^2 + (\mathbf{D} \cdot \mathbf{V})^2 + (\mathbf{D} \cdot \mathbf{W})^2 = |\mathbf{D}|^2 \\ &\mathbf{U}\mathbf{U}^{\mathsf{T}} + \mathbf{V}\mathbf{V}^{\mathsf{T}} + \mathbf{W}\mathbf{W}^{\mathsf{T}} = I \end{aligned} \tag{5}$$

where I is the 3×3 identity matrix. The minimum projection value is similarly derived,

$$\lambda_{\min} = \mathbf{D} \cdot \mathbf{C} - r\sqrt{|\mathbf{D}|^2 - (\mathbf{D} \cdot \mathbf{W})^2} - (h/2)|\mathbf{D} \cdot \mathbf{W}|$$

4 Separating Axis Tests for Two Cylinders

Given two cylinders with centers $\mathbf{C}_i$, axis directions $\mathbf{W}_i$, radii r_i, and heights h_i, for $i = 0, 1$, the cylinders are separated if there exists a nonzero direction $\mathbf{D}$ such that either

$$\mathbf{D} \cdot \mathbf{C}_0 - r_0\sqrt{|\mathbf{D}|^2 - (\mathbf{D} \cdot \mathbf{W}_0)^2} - (h_0/2)|\mathbf{D} \cdot \mathbf{W}_0| \geq \mathbf{D} \cdot \mathbf{C}_1 + r_1\sqrt{|\mathbf{D}|^2 - (\mathbf{D} \cdot \mathbf{W}_1)^2} + (h_1/2)|\mathbf{D} \cdot \mathbf{W}_1|$$

or

$$\mathbf{D} \cdot \mathbf{C}_0 + r_0\sqrt{|\mathbf{D}|^2 - (\mathbf{D} \cdot \mathbf{W}_0)^2} + (h_0/2)|\mathbf{D} \cdot \mathbf{W}_0| \leq \mathbf{D} \cdot \mathbf{C}_1 - r_1\sqrt{|\mathbf{D}|^2 - (\mathbf{D} \cdot \mathbf{W}_1)^2} - (h_1/2)|\mathbf{D} \cdot \mathbf{W}_1|$$

These are just a restatement of Equation (2) for bounded cylinders.

Defining $\mathbf{\Delta} = \mathbf{C}_1 - \mathbf{C}_0$, these tests can be combined into a single expression

$$f(\mathbf{D}) = r_0|P_0\mathbf{D}| + r_1|P_1\mathbf{D}| + (h_0/2)|\mathbf{D} \cdot \mathbf{W}_0| + (h_1/2)|\mathbf{D} \cdot \mathbf{W}_1| - |\mathbf{D} \cdot \mathbf{\Delta}| \leq 0 \tag{6}$$

where $P_i = I - \mathbf{W}_i\mathbf{W}_i^{\mathsf{T}}$ for $i = 0, 1$ are projection matrices.

If $\mathbf{\Delta} = 0$, then $f > 0$. This is geometrically obvious because two cylinders with the same center always intersect. The remainder of the discussion assumes $\mathbf{\Delta} \neq \mathbf{0}$. If $\mathbf{D}$ is perpendicular to $\mathbf{\Delta}$, then $f(\mathbf{D}) > 0$. This shows that any line perpendicular to the line containing the two cylinder centers can never be a separating axis. This is also clear geometrically. The line of sight $\mathbf{C}_0 + s\mathbf{\Delta}$ intersects both cylinders at their centers. If you project the two cylinders onto the plane $\mathbf{\Delta} \cdot (\mathbf{X} - \mathbf{C}_0) = 0$, both regions of projection overlap. No matter which line you choose containing $\mathbf{C}_0$ in this plane, the line intersects both projection regions.

If $\mathbf{D}$ is a separating direction, then $f(\mathbf{D}) \leq 0$. Observe that $f(t\mathbf{D}) = tf(\mathbf{D})$ for $t > 0$, so $f(t\mathbf{D}) \leq 0$. This is consistent with the geometry of the problem. Any nonzero multiple of a separating direction must itself be a separating direction. This allows us to restrict our attention to the unit sphere, $|\mathbf{D}| = 1$. Function f is continuous on the unit sphere, a compact set, so f must attain its minimum at some point on the sphere. This is a minimization problem in two dimensions, but the spherical geometry complicates the analysis somewhat. As we will see later, different restrictions on the set of potential separating directions can be made that yield minimization problems in a plane rather than on a sphere.

The analysis of f involves computing its derivatives to determine its critical points. These are points for which the derivative is zero or undefined. The latter category is easy to specify. The derivatives are undefined when any of the terms inside the five absolute value signs is zero. Thus, the derivatives are undefined at $\mathbf{W}_0$, $\mathbf{W}_1$, at vectors that are perpendicular to $\mathbf{W}_0$, at vectors that are perpendicular to $\mathbf{W}_1$, and at vectors that are perpendicular to $\mathbf{\Delta}$. I already argued that $f > 0$ for vectors perpendicular to $\mathbf{\Delta}$, so we may ignore this case.

The next two sections describe how to handle those directions for which the derivatives of f are undefined. The section after those describes how to handle those directions for which the derivatives of f are defined.

5 Separation Tests Involving the Cylinder Axis Directions

The cylinder axis directions themselves can be tested first for separation. The tests for separation are

$$f(\mathbf{W}_0) = r_1|\mathbf{W}_0 \times \mathbf{W}_1| + (h_0/2) + (h_1/2)|\mathbf{W}_0 \cdot \mathbf{W}_1| - |\mathbf{W}_0 \cdot \mathbf{\Delta}| \le 0$$

and

$$f(\mathbf{W}_1) = r_0|\mathbf{W}_0 \times \mathbf{W}_1| + (h_0/2)|\mathbf{W}_0 \cdot \mathbf{W}_1| + (h_1/2) - |\mathbf{W}_1 \cdot \mathbf{\Delta}| \le 0$$

If either condition is satisfied, the cylinders are separated.

The test for direction $\mathbf{W}_0 \times \mathbf{W}_1$ does not require many more operations and might lead to a quick no-intersection test,

$$f(\mathbf{W}_0 \times \mathbf{W}_1) = (r_0 + r_1)|\mathbf{W}_0 \times \mathbf{W}_1| - |\mathbf{W}_0 \times \mathbf{W}_1 \cdot \mathbf{\Delta}| \le 0$$

assuming that $\mathbf{W}_0 \times \mathbf{W}_1 \neq \mathbf{0}$. This vector is one for which the derivatives of f are undefined.

If $\mathbf{W}_0$ and $\mathbf{W}_1$ are parallel, then $\mathbf{W}_0 \times \mathbf{W}_1 = \mathbf{0}$, $|\mathbf{W}_0 \cdot \mathbf{W}_1| = 1$, and $|\mathbf{W}_0 \times \mathbf{W}_1|^2 = 1 - (\mathbf{W}_0 \cdot \mathbf{W}_1)^2$. The test for separation by $\mathbf{W}_0$ is

$$f(\mathbf{W}_0) = (h_0 + h_1)/2 - |\mathbf{W}_0 \cdot \mathbf{\Delta}| \le 0$$

If $f(\mathbf{W}_0) \ge 0$, the two cylinders are potentially separated by a direction that is perpendicular to $\mathbf{W}_0$. Geometrically it is enough to determine whether or not the circles of projection of the cylinders onto the plane $\mathbf{W}_0 \cdot \mathbf{X} = \mathbf{0}$ intersect. These circles are disjoint if and only if the length of the projection of $\mathbf{\Delta}$ onto that plane is larger than the sum of the radii of the circles. The projection of $\mathbf{\Delta}$ is $\mathbf{\Delta} - (\mathbf{W}_0 \cdot \mathbf{\Delta})\mathbf{W}_0$ and the separation test is

$$|\mathbf{\Delta} - (\mathbf{W}_0 \cdot \mathbf{\Delta})\mathbf{W}_0| \ge r_0 + r_1$$

In fact, this test is equivalent to the separating axis test for the direction $\mathbf{D} = \mathbf{\Delta} - (\mathbf{W}_0 \cdot \mathbf{\Delta})\mathbf{W}_0$. The test has a common factor $|\mathbf{D}|$ that may be divided out of the expression.

For the remainder of this document I assume that $\mathbf{W}_0$ and $\mathbf{W}_1$ are not parallel.

6 Separation Tests Involving the Cylinder Axis Perpendiculars

We now consider the directions $\mathbf{D}$ perpendicular to $\mathbf{W}_0$. The function of Equation (6) becomes

$$f(\mathbf{D}) = r_0|\mathbf{D}| + r_1|P_1\mathbf{D}| + (h_1/2)|\mathbf{D} \cdot \mathbf{W}_1| - |\mathbf{D} \cdot \mathbf{\Delta}| \tag{7}$$

We may choose $\mathbf{D}(\theta) = (\cos\theta)\mathbf{U}_0 + (\sin\theta)\mathbf{V}_0$, where $\{\mathbf{U}_0, \mathbf{V}_0, \mathbf{W}_0\}$ is a right-handed orthonormal set, giving us a circle of directions to analyze. We may then define $g(\theta) = f(\mathbf{D}(\theta))$. The analysis of the derivative of g in order to determine critical points is complicated by the presence of the four absolute value signs. We discussed previously the cases when the derivative is undefined (in terms of f itself). To compute symbolically the values θ where $g'(\theta) = 0$ is an intractable task.

An alternative parameterization is the following and is equivalent to testing directions on a hemicircle. It is sufficient to consider only a hemicircle because $-\mathbf{D}$ is a separating direction if and only if $\mathbf{D}$ is a separating direction. By assumption, $\mathbf{W}_0$ and $\mathbf{W}_1$ are not parallel. A normalized projection of $\mathbf{W}_1$ onto the plane perpendicular to $\mathbf{W}_0$ is

$$\mathbf{V}_0 = \frac{\mathbf{W}_1 - (\mathbf{W}_0 \cdot \mathbf{W}_1)\mathbf{W}_0}{|\mathbf{W}_1 - (\mathbf{W}_0 \cdot \mathbf{W}_1)\mathbf{W}_0|}$$

Define $\mathbf{U}_0 = \mathbf{V}_0 \times \mathbf{W}_0$. The set $\{\mathbf{U}_0, \mathbf{V}_0, \mathbf{W}_0\}$ is a right-handed orthonormal basis. Notice also that

$$\mathbf{U}_0 = \frac{\mathbf{W}_1 \times \mathbf{W}_0}{|\mathbf{W}_1 \times \mathbf{W}_0|}$$

Define $\mathbf{D}(t) = (1-t)\mathbf{U}_0 + t\mathbf{V}_0$ for $t \in [0,1]$, which are nonzero vectors whose normalized values cover one quarter of a circle. Define

$$F(t) = f(\mathbf{D}(t)) = r_0\sqrt{(1-t)^2 + t^2} + r_1\sqrt{(1-t)^2 + c_1^2 t^2} + h_1 b_1 t/2 - |(1-t)a_2 + tb_2| \tag{8}$$

where $a_1 = \mathbf{W}_1 \cdot \mathbf{U}_0 = 0$, $b_1 = \mathbf{W}_1 \cdot \mathbf{V}_0 > 0$, $c_1 = \mathbf{W}_1 \cdot \mathbf{W}_0$, $a_2 = \mathbf{\Delta} \cdot \mathbf{U}_0$, and $b_2 = \mathbf{\Delta} \cdot \mathbf{V}_0$. Observe that the absolute value signs were discarded for the term corresponding to $|\mathbf{D} \cdot \mathbf{W}_1|$. This is a result of knowing that the directions $\mathbf{D}$ for the hemicircle form an acute angle with $\mathbf{W}_1$, so $\mathbf{D} \cdot \mathbf{W}_1 \geq 0$. I have also used the fact that $1 = a_1^2 + b_1^2 + c_1^2 = b_1^2 + c_1^2$, in which case $1 - b_1^2 = c_1^2$.

The first derivative of $F(t)$ is

$$F'(t) = \frac{r_0(2t-1)}{\sqrt{(1-t)^2 + t^2}} + \frac{r_1((1+c_1^2)t - 1)}{\sqrt{(1-t)^2 + c_1^2 t^2}} + h_1 b_1/2 - (b_2 - a_2)\text{Sign}((1-t)a_2 + tb_2) \tag{9}$$

where $\text{Sign}(x) = +1$ when $x > 0$, -1 when $x < 0$, and 0 when $x = 0$. The second derivative is

$$F''(t) = \frac{3r_0}{((1-t)^2 + t^2)^{3/2}} + \frac{r_1 c_1^2}{((1-t)^2 + c_1^2 t^2)^{3/2}} \tag{10}$$

The warning is that $F'(t)$ and $F''(t)$ are discontinuous when $\text{Sign}((1-t)a_2 + b_2) = 0$. Notice that $F''(t) > 0$, so $F(t)$ is a *strictly convex function*. One implication is that $F'(t)$ is a strictly increasing function. Another is that $F(t)$ must have a global minimum that occurs for exactly one t on the interval $[0,1]$. The minimum is either $F(0)$, $F(1)$, or $F(\bar{t})$ where $F'(\bar{t})$ is zero or undefined. If the global minimum is nonpositive, we have a separating direction.

The algorithm is as follows. If $F(0) \leq 0$, then $\mathbf{U}_0$ is a separating direction. If $F(1) \leq 0$, then $\mathbf{V}_0$ is a separating direction. Otherwise, $F(0) > 0$ and $F(1) > 0$. If $F'(0) \geq 0$, then the convexity of F guarantees that $F(0)$ is the global minimum. Since $F(0) > 0$, there is no separation. If $F'(1) \leq 0$, then the convexity of F guarantees that $F(1)$ is the global minimum. Since $F(1) > 0$, there is no separation. We are now at the point where $F(0) > 0$, $F(1) > 0$, $F'(0) < 0$, and $F'(1) > 0$. A global minimum must occur at an interior point $\bar{t} \in [0,1]$ for which $F'(\bar{t})$ is zero or undefined. Use bisection based on the signs of $F'(t)$ to locate $\bar{t}$. At each iterate t_i it is worthwhile to test whether $F(t_i) \leq 0$. For if it is, we have found a separating direction. If the bisection terminates and $F(\bar{t}) > 0$, there is no separation.

The other line segment of directions to process has $\mathbf{D}(t) = (1-t)(-\mathbf{U}_0) + t\mathbf{V}$. The only thing that changes in Equations (8), (9), and (10) is that a_2 is replaced by $-a_2$. The bisection algorithm is applied once again to compute the global minimum of F and test whether or not F is nonpositive at the iterates.

The same algorithm is applied when the two cylinders reverse roles.

7 Separation Tests Involving Other Directions

The idea is similar to that of Section 6, to compute global minima of functions and determine whether any of the bisection iterates produces a separating direction.

The symmetry $f(-\mathbf{D}) = f(\mathbf{D})$ implies that we only need to analyze f on a hemisphere; the other hemisphere values are determined automatically. Since $f > 0$ on the great circle of vectors that are perpendicular to $\mathbf{\Delta}$, we can restrict our attention to the hemisphere whose pole is $\mathbf{W} = \mathbf{\Delta}/|\mathbf{\Delta}|$. Choose $\mathbf{U}$ and $\mathbf{V}$ such that the set $\{\mathbf{U}, \mathbf{V}, \mathbf{W}\}$ is right-handed and orthonormal.

The hemisphere is processed by decomposing it into octants and analyzing a function that is defined on a triangular surface in each octant. For example, in the first octant define $\mathbf{D}(s,t) = s\mathbf{U} + t\mathbf{V} + (1-s-t)\mathbf{W}$ and $G(s,t) = f(\mathbf{D}(s,t))$ for $s \ge 0$, $t \ge 0$, and $s + t \le 1$,

$$\begin{aligned} G(s,t) &= r_0\sqrt{s^2+t^2+(1-s-t)^2 - (a_0 s + b_0 t + c_0(1-s-t))^2} \\ &\quad + r_1\sqrt{s^2+t^2+(1-s-t)^2 - (a_1 s + b_1 t + c_1(1-s-t))^2} \\ &\quad + (h_0/2)|a_0 s + b_0 t + c_0(1-s-t)| \\ &\quad + (h_1/2)|a_1 s + b_1 t + c_1(1-s-t)| \\ &\quad - (1-s-t)|\mathbf{\Delta}| \end{aligned} \tag{11}$$

The constants a_i, b_i, and c_i are based on the following geometric observations. Because $\{\mathbf{U}, \mathbf{V}, \mathbf{W}\}$ is an orthonormal basis, we can represent $\mathbf{W}_i = a_i\mathbf{U} + b_i\mathbf{V} + c_i\mathbf{W}$, where $a_i = \mathbf{U}\cdot\mathbf{W}_i$, $b_i = \mathbf{V}\cdot\mathbf{W}_i$, and $c_i = \mathbf{W}\cdot\mathbf{W}_i$. The projections of $\mathbf{D}$ onto the cylinder axis directions are $\mathbf{D}\cdot\mathbf{W}_i = a_i s + b_i t + c_i(1-s-t)$. The projections of $\mathbf{D}$ onto planes perpendicular to the cylinder axes are $P_i\mathbf{D}$. Therefore, $\mathbf{D} = P_i\mathbf{D} + (\mathbf{D}\cdot\mathbf{W}_i)\mathbf{W}_i$, which leads to

$$\begin{aligned} |P_i\mathbf{D}|^2 &= |\mathbf{D} - (\mathbf{D}\cdot\mathbf{W}_i)\mathbf{W}_i|^2 \\ &= |\mathbf{D}|^2 - (\mathbf{D}\cdot\mathbf{W}_i)^2 \\ &= (s^2+t^2+(1-s-t)^2) - (a_i s + b_i t + c_i(1-s-t))^2 \end{aligned}$$

It turns out that $G(s,t)$ is a convex function. To simplify the presentation, define

$$L_i(s,t) = a_i s + b_i t + c_i(1-s-t)$$

and

$$Q_i(s,t) = s^2 + t^2 + (1-s-t)^2 - (a_i s + b_i t + c_i(1-s-t))^2$$

for $i = 0, 1$. The first-order derivatives are

$$\begin{aligned} Q_{i,s} &= 2s - 2(1-s-t) - 2(a_i - c_i)(a_i s + b_i t + c_i(1-s-t)) \\ Q_{i,t} &= 2t - 2(1-s-t) - 2(b_i - c_i)(a_i s + b_i t + c_i(1-s-t)) \end{aligned}$$

and the second-order derivatives are

$$\begin{aligned} Q_{i,ss} &= 4 - 2(a_i - c_i)^2 \\ Q_{i,st} &= 2 - 2(a_i - c_i)(b_i - c_i) \\ Q_{i,tt} &= 4 - 2(b_i - c_i)^2 \end{aligned}$$

The function G becomes

$$G(s,t) = r_0\sqrt{Q_0} + r_1\sqrt{Q_1} + (h_0/2)|L_0| + (h_1/2)|L_1| - (1-s-t)|\mathbf{\Delta}|$$

The first-order derivatives are

$$\begin{aligned} G_s &= r_0 \tfrac{Q_{0,s}}{2\sqrt{Q_0}} + r_1 \tfrac{Q_{1,s}}{2\sqrt{Q_1}} + (h_0/2)(a_0 - c_0)\text{Sign}(L_0) + (h_1/2)(a_1 - c_1)\text{Sign}(L_1) + |\mathbf{\Delta}| \\ G_t &= r_0 \tfrac{Q_{0,t}}{2\sqrt{Q_0}} + r_1 \tfrac{Q_{1,t}}{2\sqrt{Q_1}} + (h_0/2)(b_0 - c_0)\text{Sign}(L_0) + (h_1/2)(b_1 - c_1)\text{Sign}(L_1) + |\mathbf{\Delta}| \end{aligned}$$

The second-order derivatives are

$$\begin{aligned} G_{ss} &= r_0 \tfrac{2Q_0Q_{0,ss} - Q_{0,s}Q_{0,s}}{4Q_0^{3/2}} + r_1 \tfrac{2Q_1Q_{1,ss} - Q_{1,s}Q_{1,s}}{4Q_1^{3/2}} \\ G_{st} &= r_0 \tfrac{2Q_0Q_{0,st} - Q_{0,s}Q_{0,t}}{4Q_0^{3/2}} + r_1 \tfrac{2Q_1Q_{1,st} - Q_{1,s}Q_{1,t}}{4Q_1^{3/2}} \\ G_{tt} &= r_0 \tfrac{2Q_0Q_{0,tt} - Q_{0,t}Q_{0,t}}{4Q_0^{3/2}} + r_1 \tfrac{2Q_1Q_{1,tt} - Q_{1,t}Q_{1,t}}{4Q_1^{3/2}} \end{aligned}$$

where it is understood that the first- and second-order derivatives are discontinuous when $\text{Sign}(L_i) = 0$. If we define

$$\eta_i(s,t) = \sqrt{Q_i(s,t)}$$

then

$$G_{ss} = r_0\eta_{0,ss} + r_1\eta_{1,ss}, \;\; G_{st} = r_0\eta_{0,st} + r_1\eta_{1,st}, \;\; G_{tt} = r_0\eta_{0,tt} + r_1\eta_{1,tt}$$

The factorizations in the following were computed with the help of Mathematica,

$$Q_i^{3/2}\eta_{i,ss} = (2Q_iQ_{i,ss} - Q_{i,s}Q_{i,s})/4 = ((a_i + b_i + c_i)t - b_i)^2 \geq 0$$

and

$$Q_i^{3/2}\eta_{i,tt} = (2Q_iQ_{i,tt} - Q_{i,t}Q_{i,t})/4 = ((a_i + b_i + c_i)s - a_i)^2 \geq 0$$

These establish the facts that $G_{ss} \geq 0$ and $G_{tt} \geq 0$. The more challenging work was to factor $G_{ss}G_{tt} - G_{st}^2$. It was demonstrated that

$$\begin{aligned} Q_i^3(\eta_{i,ss}\eta_{i,tt} - \eta_{i,st}^2) &= (2Q_iQ_{i,ss} - Q_{i,s}Q_{i,s})(2Q_iQ_{i,tt} - Q_{i,t}Q_{i,t}) - (2Q_iQ_{i,st} - Q_{i,s}Q_{i,t})^2 \\ &= (a_i^2 + b_i^2 + c_i^2 - 1)p_i(s,t) \\ &= 0 \end{aligned}$$

where $p_i(s,t)$ are quadratic polynomials. However, we know that (a_i, b_i, c_i) are the coefficients of unit-length vectors $\mathbf{W}_i$ with respect to the orthonormal basis $\{\mathbf{U}, \mathbf{V}, \mathbf{W}\}$, so $a_i^2 + b_i^2 + c_i^2 = 1$. The determinant of the second-derivative matrix for G is

$$\begin{aligned} G_{ss}G_{tt} - G_{st}^2 &= (r_0\eta_{0,ss} + r_1\eta_{1,ss})(r_0\eta_{0,tt} + r_1\eta_{1,tt}) - (r_0\eta_{0,st} + r_1\eta_{1,st})^2 \\ &= r_0^2(\eta_{0,ss}\eta_{0,tt} - \eta_{0,st}^2) + r_1^2(\eta_{1,ss}\eta_{1,tt} - \eta_{1,st}^2) + r_0r_1(\eta_{0,ss}\eta_{1,tt} + \eta_{1,ss}\eta_{0,tt} - 2\eta_{0,st}\eta_{1,st}) \\ &= r_0r_1(\eta_{0,ss}\eta_{1,tt} + \eta_{1,ss}\eta_{0,tt} - 2\eta_{0,st}\eta_{1,st}) \end{aligned}$$

Finally, we were able to demonstrate that

$$Q_0^{3/2}Q_1^{3/2}(\eta_{0,ss}\eta_{1,tt} + \eta_{1,ss}\eta_{0,tt} - 2\eta_{0,st}\eta_{1,st}) = (\alpha s + \beta t + \gamma(1 - s - t))^2$$

where $\alpha = b_0c_1 - b_1c_0$, $\beta = a_1c_0 - a_0c_1$, and $\gamma = a_0b_1 - a_1b_0$. Observe that $\mathbf{W}_0 \times \mathbf{W}_1 = \alpha\mathbf{U} + \beta\mathbf{V} + \gamma\mathbf{W}$ and $\mathbf{D} \cdot \mathbf{W}_0 \times \mathbf{W}_1 = \alpha s + \beta t + \gamma(1 - s - t)$. This establishes the fact that $G_{ss}G_{tt} - G_{st}^2 \geq 0$ with equality occurring only when $\mathbf{D}$ is perpendicular to $\mathbf{W}_0 \times \mathbf{W}_1$.

The conditions $G_{ss} \geq 0$, $G_{tt} \geq 0$, and $G_{ss}G_{tt} - G_{st}^2 \geq 0$ are sufficient to conclude that $G(s,t)$ is a convex function. Numerical minimizers tend to perform quite well for convex functions.

8 Pseudocode for the Algorithm

This section describes the high-level details for the test-intersection query between two bounded cylinders.

At the top-most level, the separation function is the following. It returns true whenever the cylinders are separated.

```
bool SeparatedCylinders(
    Point C0, Vector W0, Real r0, Real h0,
    Point C1, Vector W1, Real r1, Real h1)
{
    Vector Delta = C1 - C0;
    Vector W0xW1 = Cross(W0,W1);
    Real lenW0xW1 = |W0xW1|, h0Div2 = h0/2, h1Div2 = h1/2, rSum = r0 + r1;

    if (lenW0xW1 > 0)
    {
        // Test for separation by W0.
        if (r1*lenW0xW1 + h0Div2 + h1Div2*|Dot(W0,W1)| - |Dot(W0, Delta)| < 0) return true;

        // Test for separation by W1.
        if (r0*lenW0xW1 + h0Div2*|Dot(W0,W1)| + h1Div2 - |Dot(W1, Delta)| < 0) return true;

        // Test for separation by W0xW1.
        if (rSum*lenW0xW1 - |Dot(W0xW1, Delta)| < 0) return true;

        // Test for separation by directions perpendicular to W0.
        if (SeparatedByCylinderPerpendiculars(C0,W0,r0,h0,C1,W1,r1,h1)) return true;

        // Test for separation by directions perpendicular to W1.
        if (SeparatedByCylinderPerpendiculars(C1,W1,r1,h1,C0,W0,r0,h0)) return true;

        // Test for separation by other directions.
        if (SeparatedByOtherDirections(W0,r0,h0,W1,r1,h1,Delta)) return true;
    }
    else
    {
        // Test for separation by height.
        if (h0Div2 + h1Div2 - |Dot(W0, Delta)| < 0) return true;

        // Test for separation radially.
        if (rSum - |Delta - Dot(W0, Delta)*W0| < 0) return true;

        // If parallel cylinders are not separated by height or radial distance,
        // then the cylinders must overlap.
    }

    return false;
}
```

Processing of directions perpendicular to the cylinder axes is handled by the following code.

```
Real F(Real t, Real r0, Real r1, Real h1b1Div2, Real c1sqr, Real a2, Real b2)
{
    Real omt = 1 - t;
    Real tsqr = t*t;
    Real omtsqr = omt*omt;
    Real term0 = r0*sqrt(omtsqr + tsqr);
    Real term1 = r1*sqrt(omtsqr + c1sqr*tsqr);
    Real term2 = h1b1Div2*t;
    Real term3 = |omt*a2 + t*b2|;
    return term0 + term1 + term2 - term3;
}

Real FDer(Real t, Real r0, Real r1, Real h1b1Div2, Real c1sqr, Real a2, Real b2)
{
    Real omt = 1 - t;
    Real tsqr = t*t;
    Real omtsqr = omt*omt;
```

```
    Real term0 = r0*(2*t-1)/sqrt(omtsqr + tsqr);
    Real term1 = r1*((1+c1sqr)*t - 1)/sqrt(omtsqr + c1sqr*tsqr);
    Real term2 = h1b1Div2;
    Real term3 = (b2 - a2)*sign(omt*a2 + t*b2);
    return term0 + term1 + term2 - term3;
}

bool SeparatedByCylinderPerpendiculars(
    Point C0, Vector W0, Real r0, Real h0,
    Point C1, Vector W1, Real r1, Real h1)
{
    Vector Delta = C1 - C0;
    Real c1 = Dot(W0,W1);
    Real b1 = sqrt(1 - c1*c1);
    Vector V0 = (W1 - c1*W0)/b1;
    Vector U0 = Cross(V0,W0);
    Real h1b1Div2 = h1*b1/2;
    Real c1Sqr = c1*c1;
    Real a2 = Dot(Delta,U0);
    Real b2 = Dot(Delta,V0);

    // Test directions (1-t)*U0 + t*V0.
    if (F(0,r0,r1,h1b1Div2,c1sqr,a2,b2) <= 0)
    {
        // U0 is a separating direction
        return true;
    }

    if (F(1,r0,r1,h1b1Div2,c1sqr,a2,b2) <= 0)
    {
        // V0 is a separating direction
        return true;
    }

    if (FDer(0,r0,r1,h1b1Div2,c1sqr,a2,b2) >= 0)
    {
        // no separation by perpendicular directions
        return false;
    }

    if (FDer(1,r0,r1,h1b1Div2,c1sqr,a2,b2) <= 0)
    {
        // no separation by perpendicular directions
        return false;
    }

    // Use bisection to locate t-bar for which F(t-bar) is a minimum.  The upper
    // bound maxIterations may be chosen to guarantee a specified number of digits
    // of precision in the t-variable.
    Real t0, t1, fd0, fd1, tmid, fdmid;
    int i;
    t0 = 0;
    t1 = 1;
    for (i = 0; i < maxIterations; ++i)
    {
        tmid = 0.5*(t0 + t1);
        if (F(tmid,r0,r1,h1b1Div2,c1sqr,a2,b2) <= 0)
        {
            // (1-t)*U0 + t*V0 is a separating direction
            return true;
        }

        fdmid = FDer(tmid,r0,r1,h1b1Div2,c1sqr,a2,b2);
        if (fdmid > 0)
        {
            t1 = tmid;
        }
        else if (fdmid < 0)
        {
            t0 = tmid;
        }
        else
```

```
            {
                break;
            }
        }

        // Test directions (1-t)*(-U0) + t*V0.
        a2 = -a2;
        if (F(0,r0,r1,h1b1Div2,c1sqr,a2,b2) <= 0)
        {
            // U0 is a separating direction
            return true;
        }

        if (F(1,r0,r1,h1b1Div2,c1sqr,a2,b2) <= 0)
        {
            // V0 is a separating direction
            return true;
        }

        if (FDer(0,r0,r1,h1b1Div2,c1sqr,a2,b2) >= 0)
        {
            // no separation by perpendicular directions
            return false;
        }

        if (FDer(1,r0,r1,h1b1Div2,c1sqr,a2,b2) <= 0)
        {
            // no separation by perpendicular directions
            return false;
        }

        // Use bisection to locate t-bar for which F(t-bar) is a minimum.  The upper
        // bound maxIterations may be chosen to guarantee a specified number of digits
        // of precision in the t-variable.
        t0 = 0;
        t1 = 1;
        for (i = 0; i < maxIterations; ++i)
        {
            tmid = 0.5*(t0 + t1);
            if (F(tmid,r0,r1,h1b1Div2,c1sqr,a2,b2) <= 0)
            {
                // (1-t)*U0 + t*V0 is a separating direction
                return true;
            }

            fdmid = FDer(tmid,r0,r1,h1b1Div2,c1sqr,a2,b2);
            if (fdmid > 0)
            {
                t1 = tmid;
            }
            else if (fdmid < 0)
            {
                t0 = tmid;
            }
            else
            {
                break;
            }
        }
    }
```

Processing of other directions is handled by the following code.

```
    Real G(Real s, Real t, Real r0, Real h0Div2, Real r1, Real h1Div2,
        Real a0, Real b0, Real c0, Real a1, Real b1, Real c1, Real lenDelta)
    {
        Real omsmt = 1 - s - t, ssqr = s*s, tsqr = t*t, omsmtsqr = omsmt*omsmt;
        Real temp = ssqr + tsqr + omsmtsqr;
        Real L0 = a0*s + b0*t + c0*omsmt, L1 = a1*s + b1*t + c1*omsmt;
        Real Q0 = temp - L0*L0, Q1 = temp - L1*L1;
        return r0*sqrt(Q0) + r1*sqrt(Q1) + h0Div2*|L0| + h1Div2*|L1| - omsmt*lenDelta;
```

```
}

Vector GDer(Real s, Real t, Real r0, Real h0Div2, Real r1, Real h1Div2,
    Real a0, Real b0, Real c0, Real a1, Real b1, Real c1, Real lenDelta)
{
    Real omsmt = 1 - s - t, ssqr = s*s, tsqr = t*t, omsmtsqr = omsmt*omsmt;
    Real temp = ssqr + tsqr + omsmtsqr;
    Real L0 = a0*s + b0*t + c0*omsmt, L1 = a1*s + b1*t + c1*omsmt;
    Real Q0 = temp - L0*L0, Q1 = temp - L1*L1;
    Real diffS = s - omsmt, diffT = t - omsmt;
    Real diffa0c0 = a0 - c0, diffa1c1 = a1 - c1, diffb0c0 = b0 - c0, diffb1c1 = b1 - c1;
    Real halfQ0s = diffS - diffa0c0*L0, halfQ1s = diffS - diffa1c1*L1;
    Real halfQ0t = diffT - diffb0c0*L0, halfQ1t = diffT - diffb1c1*L1;
    Real factor0 = r0/sqrt(Q0), factor1 = r1/sqrt(Q1);
    Real signL0 = sign(L0), signL1 = sign(L1);

    Vector gradient = (0,0);
    gradient[0] += halfQ0s*factor0;
    gradient[0] += halfQ1s*factor1;
    gradient[0] += h0Div2*diffa0c0*signL0;
    gradient[0] += h1Div2*diffa1c1*signL1;
    gradient[0] += lenDelta;
    gradient[1] += halfQ0t*factor0;
    gradient[1] += halfQ1t*factor1;
    gradient[1] += h0Div2*diffb0c0*signL0;
    gradient[1] += h1Div2*diffb1c1*signL1;
    gradient[1] += lenDelta;

    return gradient;
}

bool SeparatedByOtherDirections(Vector W0, Real r0, Real h0, Vector W1, Real r1, Real h1,
    Vector Delta)
{
    // Minimize G(s,t) subject to s >= 0, t >= 0, and s+t <= 1.  If at
    // any iterate you find a value for which G <= 0, return 'true'.
    // If no separating directions have been found at the end of the
    // minimization, return 'false'.
}
```

The function SeparatedByOtherDirections encapsulates a numerical method such as the Conjugate Gradient Method or Powell's Direction Set Method to minimize the function $G(s,t)$. In either case, the idea is to start at a parameter point (s_0,t_0) and choose the direction (d_0,d_1) for a line along which to minimize,

$$(s(r),t(r)) = (s_0,t_0) + r(d_0,d_1)$$

The interval for the line parameter r is determined by the intersection of the line with the triangular domain for s and t. Let this interval be denoted $[r_0,r_1]$. The function to minimize is

$$\gamma(r) = G(s(r),t(r))$$

and its derivative is

$$\gamma'(r) = G_s(s(r),t(r))d_0 + G_t(s(r),t(r))d_1 = (d_0,d_1)\cdot\nabla G(s(r),t(r))$$

Because $G(s,t)$ is a convex function, so is $\gamma(r)$. The algorithm for minimizing γ on $[r_0,r_1]$ is the same one mentioned previously for $F(t)$ on $[0,1]$. If $\gamma(r_0) \le 0$, then $\mathbf{D} = s(r_0)\mathbf{U} + t(r_0)\mathbf{V} + (1-s(r_0)-t(r_0))\mathbf{W}$ is a separating direction. If $\gamma(r_1) \le 0$, then $\mathbf{D} = s(r_1)\mathbf{U} + t(r_1)\mathbf{V} + (1-s(r_1)-t(r_1))\mathbf{W}$ is a separating direction. Otherwise, $\gamma(r_0) > 0$ and $\gamma(r_1) > 0$. If $\gamma'(r_0) \ge 0$, then the convexity of γ guarantees that $\gamma(r_0)$ is the global minimum. Since $\gamma(r_0) > 0$, there is no separation *using directions along the current r-interval*. If $\gamma'(r_1) \le 0$, then the convexity of γ guarantees that $\gamma(r_1)$ is the global minimum. Since $\gamma(r_1) > 0$, there is

no separation *using directions along the current r-interval*. Otherwise, $\gamma(r_0) > 0$, $\gamma(r_1) > 0$, $\gamma'(r_0) < 0$, and $\gamma'(r_1) > 0$. A global minimum must occur at an interior point $\bar{r} \in [r_0, r_1]$ for which $\gamma'(\bar{t})$ is zero or undefined. Use bisection based on the signs of $\gamma'(r)$ to locate $\bar{r}$. At each iterate r_i it is worthwhile to test whether $\gamma(r_i) \leq 0$. For if it is, we have found a separating direction. If the bisection terminates and $\gamma(\bar{r}) > 0$, there is no separation *using directions along the current r-interval*.

Once a minimum has been located along the current r-interval, say, $\gamma(\bar{r})$, then the new starting point is $(s_0, t_0) = (s(\bar{r}), t(\bar{r}))$ and a new direction is chosen for (d_0, d_1). If the minimum of the previous line occurs on a boundary of the triangular domain, then you need only search that boundaries of the triangular domain for the global minimum. Otherwise, the line search is repeated for the new line. If the global minimum is interior to the triangular domain, then you will never reach the boundary on a line search.

The choice of directions depends on the numerical minimizer you choose. The number of lines to search is usually based on convergence criteria: are you close enough to a global minimum?